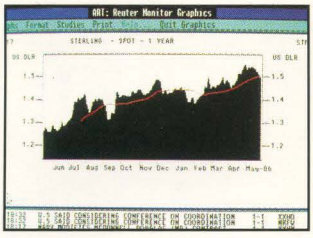
OCTOBER 1986 VOL1 NO1 MICROSOFT®

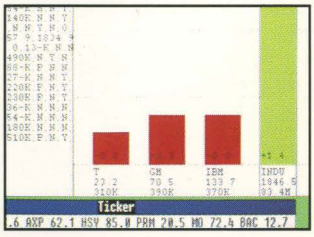
Special Windows Issue

Systems Journal



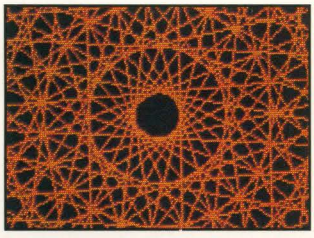
1 Advanced Reuter Terminal Gives Traders Windows on Financial World

The Advanced Reuter Terminal (ART), a dedicated hardware and software system that runs under Microsoft® Windows, enables currency traders to view and analyze up-to-the-minute financial data. An exclusive interview reveals the developing team's experiences.



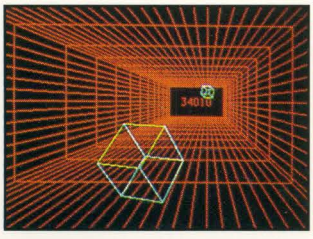
7 DDE: A Public Protocol for Advanced Application Linkages

An extension of Microsoft Windows data exchange facilities, the Dynamic Data Exchange (DDE) protocol allows developers to establish sophisticated application linkages, ranging from automatic updating of documents to processing of real-time data.



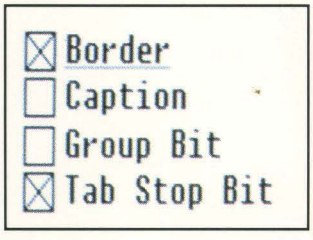
17 New Intel Graphics Coprocessor Makes Windows Sparkle

Intel® has a new way to boost the display speed of Microsoft Windows and other graphics applications: the 82786, a high-powered coprocessor that improves workstation performance, moving graphics functions from software to high-speed hardware.



21 TI's Programmable Graphics Processor Perks up PC Pixels

The first true graphics microprocessor, Texas Instruments' 32-bit TMS34010 executes at 6 MIPS, addresses up to 4 gigabytes of memory, supports displays of up to 64K by 64K pixels, and includes high-speed pixel-processing instructions.



25 Latest Dialog Editor Speeds Windows Application Development

An improved version of the Microsoft Windows Dialog Editor (supplied with the software development kit) simplifies designing and testing a dialog box—one of the key elements of a Windows application. Here's how to use it and what to watch out for.

16 GENIE—The Windows Developers RoundTable
20 Windows Training Seminars
28 DIAL—Direct Access to Product Support
29 Ask Dr. Bob

EDITOR'S NOTE

This first issue of the Microsoft Systems Journal is devoted in its entirety to Microsoft Windows operating environment, Microsoft's most important system software product since MS-DOS®.

Windows has encountered remarkable success both as a retail product and as a development environment since its release last November. While much of the product's initial popularity can be attributed to its flexibility as a "switcher" for traditional applications and the included sample applications, its ultimate success will depend on the acceptance of native Windows applications.

Reuters' ART

As a definitive example of the capabilities of Microsoft Windows and its flexibility to serve particular user interface designs, we examine the Advanced Reuter Terminal (ART), the most sophisticated Windows application to date. A superb example of design and coding craftsmanship, the ART would be worth featuring even if it were not based on Windows.

We also present profiles of two graphics processors that will have a significant impact on future graphics systems: the Intel 82786 and the Texas Instruments TMS34010. And we take a look at DDE, a new standard for interprogram data exchange, as well as examine the improved Dialog editor. In addition, Dr. Bob tackles some questions about Windows from developers. In sum, we hope to present a developer's view of Microsoft Windows.

Future Issues

In future issues we will focus on other areas of interest to developers, including C programming, device drivers, extended and expanded memory, debuggers, code portability, and the inside view of MS-DOS and Microsoft XENIX®. We will tackle the subjects that interest serious developers most—from user interface design to guidelines for writing terminate-and-stay-resident programs. *Our aim is to confront the real issues confronting serious developers.*

Write to Us

We invite you to write to us, to subscribe, and equally important, to tell us what you think we should include in the Journal. Beginning in January 1987, our issues will expand to full size, more than twice the length of this special issue.

We hope the Journal will become one of the most important publications you receive.

—Ed.

The Microsoft Systems Journal (ISSN #0889-9932) is published every other month by Microsoft Corporation, 10611 NE 36th Way, Box 97017, Redmond, Washington 98073-9717. Officers: William H. Gates, III. Chairman of the Board and Chief Executive Officer; Jon Shirley, President and Chief Operating Officer; Francis J. Gaudette, Treasurer; William Neukom, Secretary. Subscription rate: $50.00 per year (single copies $10.00). To order subscriptions or report change of address, write to the Circulation Department, Microsoft Systems Journal, Box 97017, Redmond, Washington 98073-9717. Manuscript submissions and all other correspondence should be addressed to H. Roark & Associates, Suite 804, 141 East 44th St., New York, NY 10017.
Copyright © 1986 Microsoft Corporation. All rights reserved; reproduction in part or in whole without permission is prohibited.

EDITOR **Jonathan D. Lazarus** **Michael Longacre** DESIGN

ART Showcases Microsoft Windows

1

MSJ Interview

One of the finest examples to date of software running under Microsoft Windows is a product called the ART, the Advanced Reuter Terminal. When you think of Reuters, you think of news, but to currency traders who buy and sell millions of dollars each day, Reuters means finance. Its financial information service is the lifeblood of their trade, providing the timely and accurate information on which their success depends. The ART is a complete hardware and software solution that supplies a high-resolution color monitor and features multiple windows showing text and graphics displays of up-to-the-minute financial information. It has the equivalent display capacity of ten of the previous Reuter Monitor terminals, which limited users to viewing information on a monochrome text display. What's more, ART users can perform data analyses by choosing options from pull-down menus.

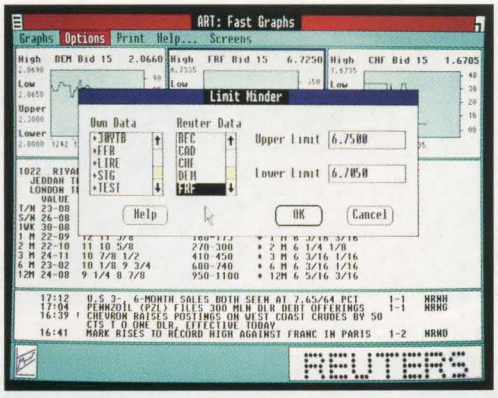
ART's Configuration

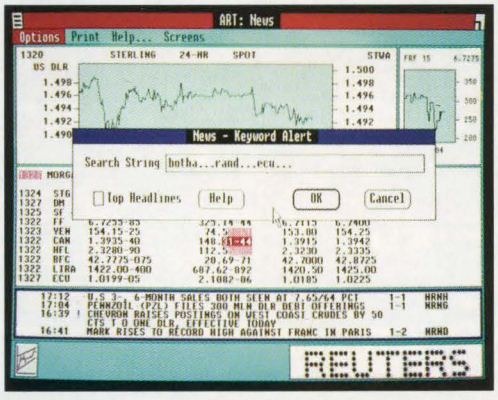
ART hardware consists of an 80286-based (PC-AT compatible) NCR computer with 1.5 megabytes of RAM, a 20-megabyte hard disk, high-resolution (EGA) display, 1200-baud modem, Reuters proprietary communications board, a Reuters programmable keyboard, and, of course, a mouse.

But the heart of the system is the Windows-based software. Developed by the Reuters office in London, it offers such capabilities as real-time graphics, user-defined montage pages, limit and news watching, and graphics analysis. Real-time graphics are made possible by calling up a graphics window and linking it to a particular currency value displayed on the screen. Each time the value of the currency changes, a new tick is added to the graph.

The ART provides automatic limit watching, using dialogs in which users enter upper and lower limits for any currency displayed on a stored page. When limits are broken, the ART gives audible and visible signals allowing a dealer to react instantly. News watching works the same way; the ART immediately alerts the user when the preselected word or phrase appears in a headline. In sum, the ART system allows the user to determine the screen's content and layout, an asset to active traders.

ART HARDWARE CONSISTS OF AN 80286-BASED NCR COMPUTER, A REUTERS PROGRAMMABLE KEYBOARD, AND, OF COURSE, A MOUSE.





THE ART PROVIDES LIMIT WATCHING, USING DIALOGS IN WHICH USERS ENTER UPPER AND LOWER LIMITS FOR ANY CURRENCY.

REUTERS: We have to go through a lot of tricks to design the system in such a way that the time-critical aspects of the application are catered to. There must be an easier way.

What's also a major problem is 640K restrictions. It's imperative that we're able to access lots of memory. For large data areas you can do something with RAM disks, but we really need MS-DOS 5.0. It seems to be receding into the far distance; every quarter it seems to slip a quarter. The whole point of a windows environment is to have multiple simultaneous applications running, and if memory limitations prevent loading more than two applications, that's really bad news.

If Microsoft is going to keep faith with the developers of Windows applications, it has to come out with MS-DOS 5.0—or at least a mechanism for making large chunks of executable memory available.

Cautious Interest

MSJ: You mentioned that in the course of developing your product you had to come up with some tricks to perform certain tasks. Do you ever share this information with other developers? Would, for instance, a user group benefit from your experiences, or would you rather keep that kind of information to yourself?

REUTERS: I'd characterize our attitude as "cautious interest." You must understand that the major users of Microsoft Windows in the U.K. are either our customers or competitors. And so, while we will certainly chat about our experiences and exchange information on bugs and so on, we would not be quite as forthcoming as perhaps a company that was trying to advertise itself.

MSJ: The keyboard used with ART is quite different from the normal PC-style keyboard. Tell us something about it.

REUTERS: The keyboard uses an 80186, which gives it more processing power than a PC. It is completely programmable, which we need since we have many different installations, which drive different kinds of terminal keyboards. The keyboard we use lets us emulate any of these.

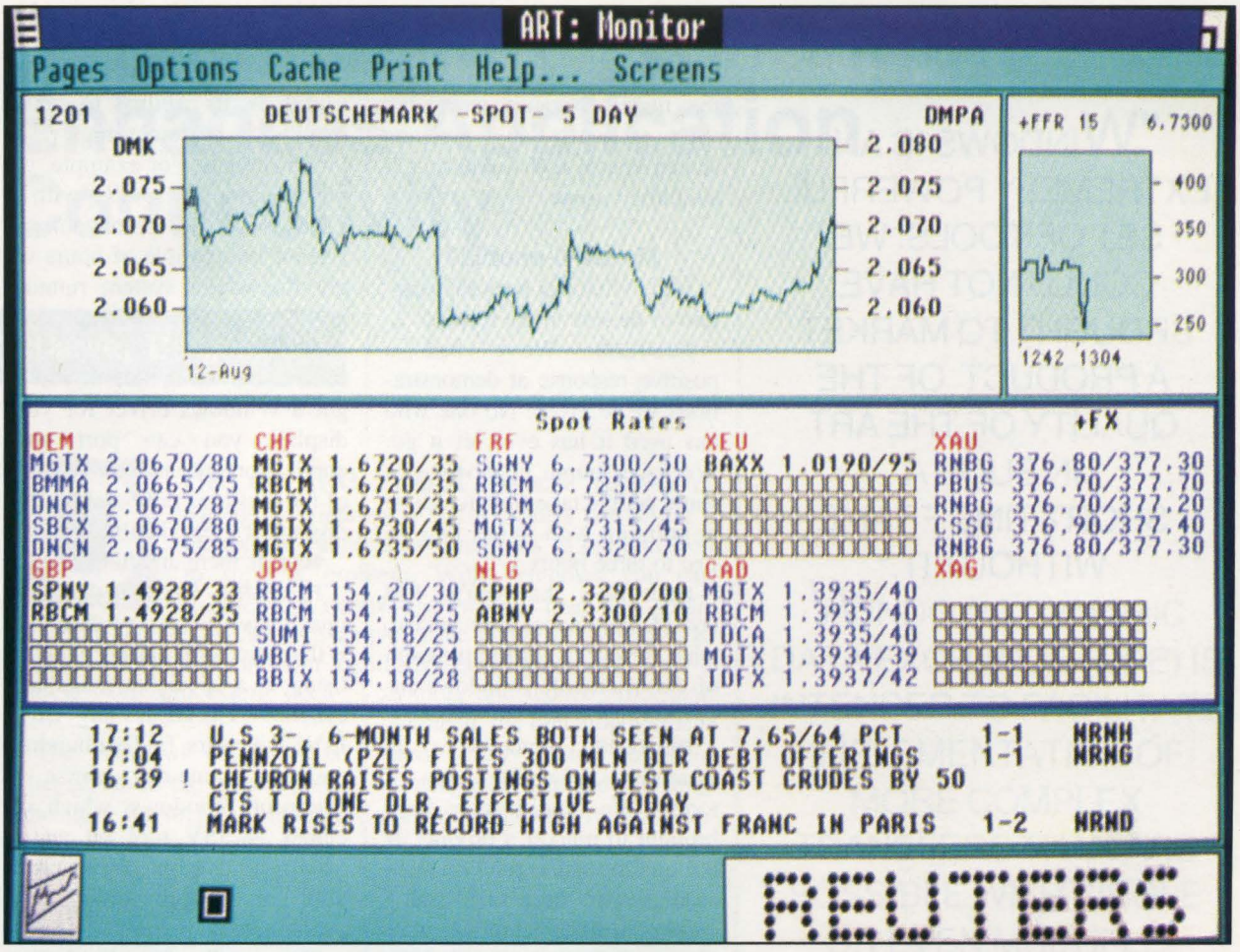
MSJ: What is the function of your proprietary communications board?

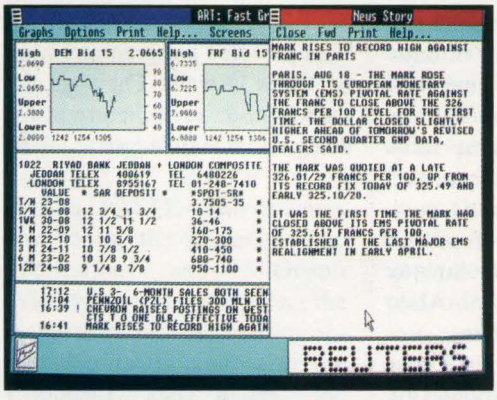
REUTERS: The board has its own processor and handles the communications link, which is a multi-drop line protocol. The board also de-couples communications processing from Windows processing and does page-caching. Information pages are put into cache memory so that dealers will get a quicker response when changing pages.

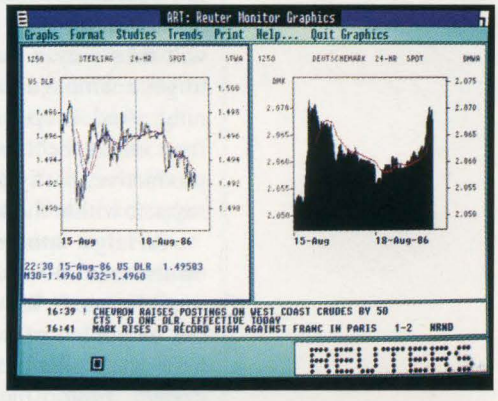
MSJ: Are you looking at any of the new graphics chips like the Intel 82786 and TI34010 to enhance the system?

REUTERS: Yes. We've developed a super-high-resolution graphics board that uses the Intel chip, and we are waiting for Intel to release the final version. We were also impressed with the TI chip, and we're looking at it seriously. We'll use the one that's up and running at board production time.

MSJ: Your windows work a little differently from the typical Microsoft windows. Why did




ART: Fast Gr
News Story
Graphs Options Print Help... Screens
Close Fwd Print Help...
High DEM Bid 15 2.0665
Low
Upper
Lower
High FRF Bid 15
Low
Upper
Lower
MARK RISES TO RECORD HIGH AGAINST FRANC IN PARIS
PARIS, AUG 18 - THE MARK ROSE THROUGH ITS EUROPEAN MONETARY SYSTEM (EMS) PIVOTAL RATE AGAINST THE FRANC TO CLOSE ABOVE THE 326 FRANCS PER 100 LEVEL FOR THE FIRST TIME. THE DOLLAR CLOSED SLIGHTLY HIGHER AHEAD OF TOMORROW'S REVISED U.S. SECOND QUARTER GNP DATA, DEALERS SAID.
THE MARK WAS QUOTED AT A LATE 326.01/29 FRANCS PER 100, UP FROM ITS RECORD FIX TODAY OF 325.49 AND EARLY 325.10/20.
IT WAS THE FIRST TIME THE MARK HAD CLOSED ABOVE ITS EMS PIVOTAL RATE OF 325.617 FRANCS PER 100, ESTABLISHED AT THE LAST MAJOR EMS REALIGNMENT IN EARLY APRIL.
1022 RIYAD BANK JEDDAH ▸ LONDON COMPOSITE
JEDDAH TELEX 400619 TEL 6480226
-LONDON TELEX 8955167 TEL 01-248-7410
VALUE * SAR DEPOSIT * *SPOT-SR*
1/W 23-08 3.7505-35 *
S/W 26-08 12 3/4 11 3/4 10-14 *
1WK 30-08 12 1/2 11 1/2 36-46 *
1 M 22-09 12 11 5/8 160-175 *
2 M 22-10 11 10 5/8 270-300 *
3 M 24-11 10 7/8 1/2 410-450 *
6 M 22-02 10 1/8 9 3/4 680-740 *
12M 24-08 9 1/4 8 7/8 950-1100 *
17:12 U.S 3- 6-MONTH SALES BOTH SEEN
17:04 PENNZOIL (PZL) FILES 300 MLN DL
16:39 ! CHEVRON RAISES POSTINGS ON WEST
CTS T O ONE DLR, EFFECTIVE TODA
16:41 MARK RISES TO RECORD HIGH AGAIN
REUTERS


ART: Reuter Monitor Graphics
Graphs Format Studies Trends Print Help... Quit Graphics
STERLING 24-HR SPOT
US DLR
15-Aug
18-Aug-86
DEUTSCHEMARK 24-HR SPOT
DMK
15-Aug
18-Aug-86
22:30 15-Aug-86 US DLR 1.49583
M30=1.4960 V32=1.4960
16:39 ! CHEVRON RAISES POSTINGS ON WEST COAST CRUDES BY 50 CTS T O ONE DLR, EFFECTIVE TODAY
16:41 MARK RISES TO RECORD HIGH AGAINST FRANC IN PARIS 1-2 HRND
REUTERS

The ART offers real-time graphics, user-defined montage pages, limit and news watching, and graphics analysis. Individual currency exchange rates are displayed in real-time graphs. Each time the value of a currency changes, a new tick is added to the graph.

"WINDOWS IS AN EXTREMELY POWERFUL SET OF TOOLS. WE COULD NOT HAVE BROUGHT TO MARKET A PRODUCT OF THE QUALITY OF THE ART IN SUCH A SHORT TIME FRAME WITHOUT IT."

you choose to have a common menu bar?

REUTERS: We eliminated the menus because we didn't want to waste any precious screen space, and found that it simplified usage.

Mouse-o-phobia?

MSJ: What has been the reaction of dealers to the mouse?

REUTERS: We've had a positive response at demonstrations we've given. No one who has used it has ever let it go. We have found, however, that some people take time to adjust to pointing with it—as long as two to three hours.

MSJ: Now that you've had considerable experience with it, what is your overall impression of the Microsoft Windows environment?

REUTERS: Windows is an extremely powerful set of tools. We could not have brought to market a product of the quality of the ART in such a relatively short time frame without it. The learning curve is reasonably short, too. An experienced C programmer will take only a matter of days to get a simple application running. And in perhaps four to five weeks he'll become fully productive, partly as a result of support within the team.

A large number of people on the team now know how to use Microsoft Windows. Also, the examples of working systems make the learning curve easier. From that point of view, I'm convinced Windows is the right choice for Reuters. As an environment for running third-party applications, Windows might not be the ideal choice, but we're using it for specialized applications within the financial marketplace.

The thing that makes it so useful is its ability to adapt quickly to different high-resolution screens. For example, we pulled in a Wyse PC with a 1280 by 786 monochrome screen. In a couple of hours we got the whole system running in a totally different resolution, different character sets, different fonts and looking nice. If you've got a Windows driver for your display, you can port your application onto it and make use of the facilities provided by that display. It's that flexible.

MSJ: Is there anything else?

REUTERS: One of the most imaginative ideas I've seen yet is that Microsoft has been advertising in a specialist magazine, called DOT.EXE. It's targeted at ISVs and offers, for four hundred pounds, a one-day course in Microsoft Windows, which includes an ISV tool kit and a Microsoft Mouse. We'll probably be plugging our new people into it.

The credit for developing ART should go to a team of high-powered programmers led by Chris Oswald. They are the ones who deserve the credit for doing an incredible job. ■

Dynamic Data Exchange Enhances Application Connectivity

7

Harvey Berger

One of the most valuable benefits of Microsoft Windows is that it enables you to use a number of application programs in combination to solve a problem. With Windows you can transfer text, numbers, or pictures from one application to another. A classic example is the ability of Windows to take a table or a graph from a spreadsheet program and insert it into a word processing document.

The Windows Clipboard is what serves as the intermediate repository for such information exchanges among applications. At your direction, information is placed on the Clipboard by a CUT or COPY operation in one application and then inserted into the context of a second application with a PASTE operation.

On the surface the process seems straightforward—a selection, a CUT or COPY, and then a PASTE—and manual. But behind the scenes the two applications and Microsoft Windows combine to effect the actual transfer. The donating application takes control of the Clipboard and places the selected information on it in one or a series of formats. The receiving application then selects a format and requests the data from the Clipboard.

In many cases, to save memory and reduce processing overhead, the donating applications may defer actually rendering the selected data until another application requests it.

A Step Forward

The Clipboard's facilities are quite flexible, but they require manual intervention for each transfer, and the Clipboard can hold only one "item" at a time. In a multitasking environment like Windows, the need rapidly develops for more sophisticated data exchange facilities. Windows Dynamic Data Exchange (DDE) is intended to facilitate implementation of more complex information transfers than are possible with single or even multiple clipboards.

DDE is a formal, public protocol for data exchange built on Windows messaging, and the shared memory facility. DDE is much more an agreement among cooperating applications than it is a partic-

WINDOWS DYNAMIC DATA EXCHANGE (DDE) IS INTENDED TO FACILITATE IMPLEMENTATION OF MORE COMPLEX TRANSFERS THAN ARE POSSIBLE WITH SINGLE OR EVEN MULTIPLE CLIPBOARDS.

Figure 1: DDE Messages

DDE Messages	Purpose	Argument 1	Argument 2
WM_DDE_INITIATE	Ask if a window will exchange items related to a topic	Atom -> Topic	
WM_DDE_TERMINATE	End the exchange of items related to the topic	Atom -> Topic	
WM_DDE_ACK	Response to a DDE message	Atom	Acknowledgement code
WM_DDE_REQUEST	Request a given item	Atom -> Item Name	Desired data format
WM_DDE_DATA	Contains a requested data item	Atom ->Item Name	Handle to data object
WM_DDE_ADVISE	Request a given item and automatic updating	Atom -> Item Name	Desired data format
WM_DDE_UNADVISE	Request end of automatic updating for a given item	Atom ->Item Name	Desired data format
WM_DDE_POKE	Contains an unsolicited data item	Atom ->Item Name	Handle to data object
WM_DDE_EXECUTE	Request execution of indicated command		Handle to data object containing command

ular body of code or executable routine. It's a special usage of Windows messages.

Windows applications routinely receive messages corresponding to user- and system-initiated actions ranging from keystrokes to requests to repaint the screen. DDE uses a set of formerly reserved message numbers to implement the protocol. In the Microsoft Windows environment any application may send a message to any other application, without explicit knowledge of the nature of the other application.

DDE is designed to facilitate implementation of a wide variety of data exchanges, ranging from simple one-time transfers of selected data to automatic parallel transfers of real-time data. Since many DDE relationships are likely to be more complicated than those of the Clipboard, DDE is described in terms of a server and client applications rather than in terms of a donor and a receiver, where a single server may be supplying data to many clients and a single client application may be communicating with many servers simultaneously.

Data Formats

DDE and the Clipboard share a number of common facilities, most notably an extensible set of data exchange formats. Initially Microsoft Windows recognized a limited set of public formats for data exchange (TEXT, BIT MAP, METAFILE, SYLK, DIF, and so forth) and custom formats. Concurrent with the announcement of DDE, Microsoft has announced five additional clipboard data formats (Rich Text, Table, Sub-Table, WKS, and DBF).

Windows also allows cooperating applications to register custom formats for private usage. The new formats allow the economical transfer of data with its formatting information.

Object Orientation

A key distinction between Clipboard and DDE exchanges is DDE's object orientation. The Clipboard is used to transfer a user selection from one application to another. Each Clipboard transfer is independent; the sequence of transfers corresponds to the sequence of user operations. With DDE each transfer consists of an item associated with a "name."

Names and Topics

When requesting or supplying data, the names are provided to allow the communicating task to identify the source or disposition of the data item. Names describe either the location or the contents of a data item. In a spreadsheet, each cell has an address (for example, R2C3 for the cell in the second row and third column). Some spreadsheets also have named cell ranges (R1:4C5, for example, might be named TOTALS). In either case, a natural relationship exists between names and data items. With other kinds of applications, like word processors, names are likely be to more positionally oriented (page 3, for example). The DDE protocol requires a name or a positional reference for each item to be transferred.

By definition, each name, as recognized by the server, refers uniquely to a specific data item. The choice of names and topics are the private concern of a server and its clients. Some servers will associate names relevant to a given subject with a topic. Client applications typically ask potential servers if they recognize (and can supply data about) the topic. Where the server's names are in themselves very complete descriptions of data items, a sample name may be used as the topic. Servers that define topics should be worded carefully to avoid accidental duplication.

Applications

In the first example the Clipboard is used to paste a chart into a word processing document. With DDE it is possible to construct more complex scenarios. In addition to saving the pasted chart, the word processor might store the name of the chart, the file name worksheet it was associated with, and the file name of the application program. Given that information, the word processor could ensure that it had the latest version of the chart.

First the word processor could, after verifying existence of the worksheet file, start executing the spreadsheet application, specifying the work-

Figure 2: Client's Window Procedure

```
typedef struct {
    WORD wDDEMODE;
    WORD wFormat;
}  DDEADVISEBLOCK;
typedef struct {
    WORD wClientRelease;
    WORD wFormat;
    WORD hServerWnd;
    WORD ClipData;
}  UPDATEINFO;
ATOM aCompare;
BOOL bAccepted;
long FAR PASCAL SampleRcvrWndProc(hWnd, message, wParam, lParam)
    switch (message)
{
                                ⋮

        case WM_DDE_ACK:
            aCompare  = HIWORD(lParam);
            bAccepted = LOWORD(lParam);
                /* Is this acknowledgement of first inquire message? */
            if ((aCompare == aStockInfo) && bAccepted)
                hwndServer = wParam;
            else {
                /* is this the acknowledgement of an advise message? */
                for (i = 0; i<NUMSTOCKS; i++) {
                    if (aCompare == stockinfo[i].Atom) {
                        /* Record Server's response to Stock Symbol */

                                ⋮

                    }
                }
            }
            break;
         case WM_DDE_DATA:
            ProcessNewData(wParam, lParam);
            break;
        default:
            return((long)DefWindowProc(hWnd, message, wParam, lParam));
            break;
    }
    return(0L);
}
```

DDE IS DESIGNED TO FACILITATE IMPLEMENTATION OF A WIDE VARIETY OF DATA EXCHANGES, RANGING FROM SIMPLE ONE-TIME TRANSFERS OF SELECTED DATA TO AUTOMATIC PARALLEL TRANSFERS OF REAL-TIME DATA.

Figure 3: Client's Window Creation Procedure

```
SampleRcvrCreate(hwnd)
HWND hwnd;
{
HMENU hMenu;
DDEADVISEBLOCK far *lpmem;
int i;
         /* DDEAddAtom is part of the global atom manager
          * in the dynamically linked library -- DDE.EXE */
aStockInfo = DDEAddAtom((LPSTR)"NYSE.COM.STK");
         /* For the first message, broadcast it to everyone;
          * When we return from this call, hwndServer will be valid
          * if a server exists.  NULL as loword of lParam means server
          * can be any application. */
SendMessage((HWND)-1, WM_DDE_INITIATE, hwnd,
                        MAKELONG(NULL, aStockInfo));
                        DDEDeleteAtom(aStockInfo);
        /* If Server isn't available, return. hwndServer
         *         can't be null in a call to PostMessage */
if (hwndServer == NULL)     return;
      /* Some Dummy Stocks for this example */
stockinfo[0].Atom = DDEAddAtom((LPSTR)"T");
stockinfo[1].Atom = DDEAddAtom((LPSTR)"GM");
stockinfo[2].Atom = DDEAddAtom((LPSTR)"IBM");
stockinfo[3].Atom = DDEAddAtom((LPSTR)"INDU");

                        ⋮

    for (i = 0; i< NUMSTOCKS; i++) {
        hmem[i] = GlobalAlloc(GHND, (DWORD)4);
        lpmem = (DDEADVISEBLOCK far *)GlobalLock(hmem[i]);
        lpmem->wDDEMODE = 0;     /* I'm not going to ACK his updates */
        lpmem->wFormat = CF_TEXT;
        GlobalUnlock(hmem[i]);
        PostMessage( (HWND) hwndServer, WM_DDE_ADVISE, hwnd,
                     MAKELONG(hmem[i],stockinfo[i].Atom)  );
    }
```

THE TERMS *CLIENT* AND *SERVER* ARE LARGLEY SEMANTIC, SINCE EITHER APPLICATION CAN SEND (OR RECEIVE) ANY MESSAGE TYPE.

sheet file name and a DDE startup parameter. Then, using DDE, the word processor could request retransfer of the graph. The exact procedures to be used are left up to the developers of the respective applications. Microsoft will most likely issue guidelines soon for future application development.

Chain Reaction

A scenario in which the chart is the result of calculations in a spreadsheet is also reasonable to envision. When printing the document, the word processor might request the chart from the graphics application, which in turn might extract the applicable data from the spreadsheet. The integration becomes involved, but the burden moves from the user repeating manual steps (which he might forget) to the PC using DDE.

DDE Sequences

A sequence of DDE exchanges between a client and a server are referred to as sessions.*(Figures 2, 3, 4, 5, 7, 8, 9 show portions of sample client and server applications.)* A session typically begins with an INITIATE message. A client application sends a WM_DDE_INITIATE to every other active application (potential server) in the system. The INITIATE message includes a topic or a sample name. If an active server is willing to exchange data related to the topic, it in turn responds with a WM_DDE_ACK message.

After establishing a session, the client and the server exchange messages according to the operating rules of the server. Servers may have implementation restrictions on the number of clients or the types of transfers it will honor. DDE provides eight message types that establish transfer protocols.

The terms *client* and *server* are largely semantic, since either application can send (or receive) any message type. In practice, a given application may be functioning as a server to one application and as a client to the same or another application.

A simple example of a DDE transfer would consist of the following sequence:

1. Application One broadcasts an INITIATE message to those windows in search of a server that will discuss CHEMICAL FORMULAS.

2. APP2, with a database of chemical formulas, responds to

APP1 with an ACK (acknowledgement) message.

3. APP1 then sends a REQUEST message specifying WATER as the desired item name.

4 APP2 responds with a text string DATA message pointing to WATER=H_2O.

5. APP1 then sends a REQUEST message for SALT.

6. APP2 responds with an DATA message specifying SALT=NaCl.

Real-Time Data

DDE was developed in response to the users' need for the coupling of data from separate applications. The stock market is another classic example of the need for DDE. During trading hours there's a constant flow of data as each transaction is processed by the exchanges. Market watchers need to filter the data stream and isolate the transactions of interest to them. The selected data is then analyzed to determine the required action. One of the first demonstrations of DDE modeled this process.

A Signal™ receiver (supplied by Lotus Information Network Corporation) offered a continuous feed of stock prices (for up to 600 selected stocks). Normally the Signal receiver (connected to a serial port) supplies data to special Lotus® software. For this demonstration Windows Terminal was modified to detect price changes and respond to DDE messages. A simple bar-charting program was then written to receive the DDE DATA messages.

Interestingly the SIGNAL monitor program was written in New York City by consultant Louis Cutrona, and the Charting program was written by Mark Taylor of Microsoft. When brought together, the two programs were able to communicate through DDE after being developed 3,000 miles apart.

Figure 4: Client's Data Update Procedure

```
ProcessNewData(hwnd, lStuff)
HWND hwnd;
DWORD lStuff;
{
    ATOM     aData    = HIWORD(lStuff);
    HANDLE   hDDEData = LOWORD(lStuff);
    UPDATEINFO far *lpDataBlock;
    LPSTR lpData;
    HDC hdc;
    int i,c;
        /* Make sure we're getting info only from our server */
        /* Note: hwndServer is a  global variable */
    if (hwnd != hwndServer)
        return;
    lpDataBlock = (UPDATEINFO far *)GlobalLock(hDDEData);
    for (i = 0; i<NUMSTOCKS; i++) {
        if (aData == stockinfo[i].Atom) {
                                  :
        }
    }
    GlobalUnlock(hDDEData);
    GlobalFree(hDDEData);
}
```

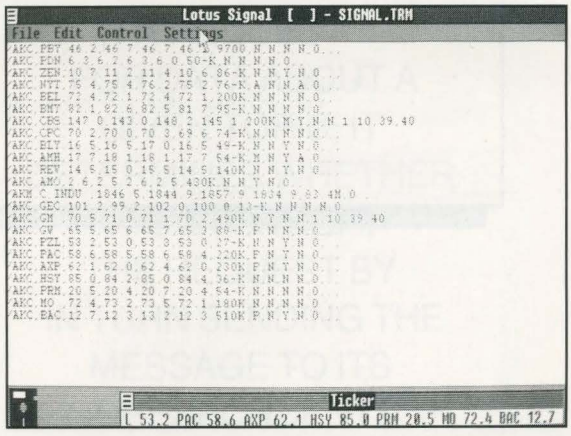
ADVISE-DATA

In the first example above, the transfers are relatively simple. For each REQUEST message there's a corresponding DATA . The ADVISE message allows for more complex transfers. In the following example it accommodates the automatic transfer of real-time data as it changes.

1. APP3 broadcasts an INITIATE message to those windows, looking for a server with NY STOCK PRICES.

2. The Signal Monitor Application Four (APP4) responds with an ACK.

3. APP3 sends APP4 an ADVISE message that specifies IBM.®



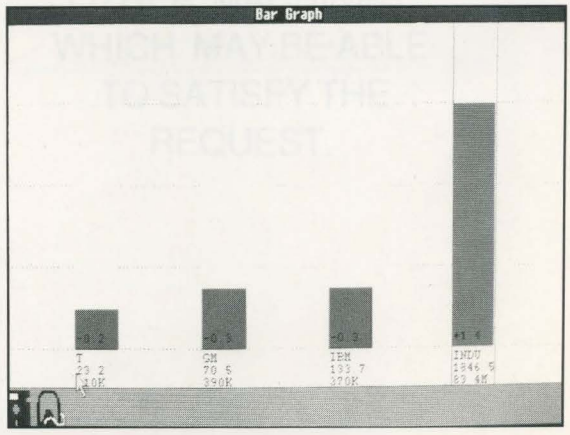


Figure 5: Server's Window Procedure

```
    /* Procedures which make up the window class. */

long FAR PASCAL TermWndProc(hwnd, message, wParam, lParam)

                                ⋮

    switch (message)
        {
                                ⋮

        case WM_DDE_INITIATE:
                dde_initiate( hwnd, wParam, lParam );
                break;
        case WM_DDE_ADVISE:
                dde_advise( hwnd, wParam, lParam );
                break;
        case WM_DDE_UNADVISE:
                dde_unadvise( hwnd, wParam, lParam );
                break;
        case WM_DDE_TERMINATE:
                dde_cancel  ( hwnd, wParam, lParam );
                break;
        case WM_DDE_REQUEST:
                dde_request( hwnd, wParam, lParam );
                break;
        case WM_DDE_EXECUTE:
                dde_execute( hwnd, wParam, lParam );
                break;

                                ⋮

        default:
            return(DefWindowProc(hwnd, message, wParam,lParam));
            break;
    }
    return(0L);
}
```

4. APP4 responds with an ACK indicating that it recognizes IBM and can supply data about it.

6. APP4 then sends the current price data for IBM.

7. Steps 3 to 5 are repeated for each stock of interest.

8. Each time APP4 learns of a new price for stock about which it has received an ADVISE, it sends an DATA message with the new price.

Atom Management

To speed processing and simplify the development of DDE Servers, DDE applications exchange "atoms" rather than strings. A central atom manager allows all applications in the system to register strings in exchange for a 16-bit atom. The atom manager (a shared executable library) will also translate the atom to its corresponding string. A single atom pool is maintained for all DDE applications in the system.

In anticipation of sophisticated client/server relationships, DDE offers optional specification of the Clipboard format for data exchange, acknowledgement of messages, and shared memory controls. Private client/-server priority schemes are also accommodated.

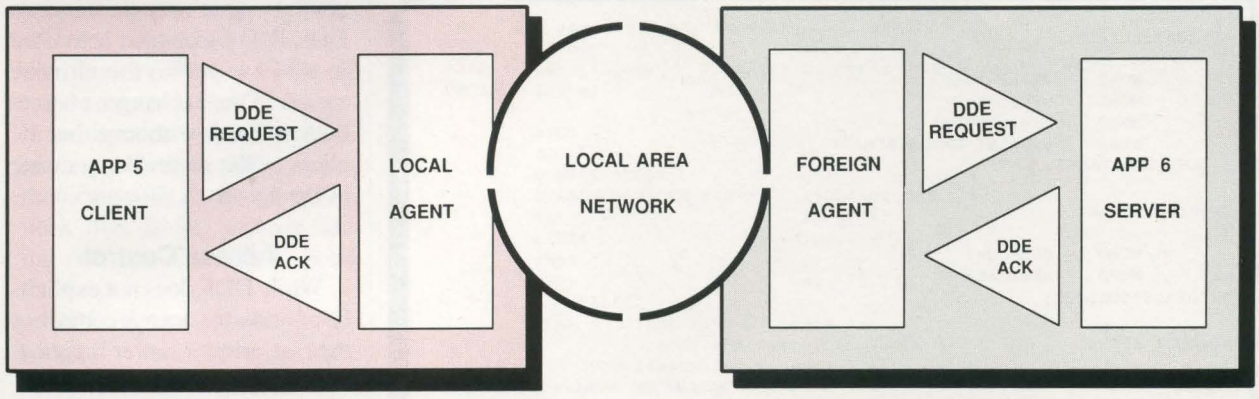
Developers interested in the details of DDE, the central atom manager, or the new Clipboard formats can download the specifications documents and sample programs from either DIAL (see page 28) or GENIE (see page 16).

On the basis of early response to the DDE design specification, a significant number of the new Microsoft Windows applications now being developed will incorporate DDE provisions. The nature of the implementation depends on the type of application and its particular features.

Macros

Many applications, including spreadsheets, databases, and charting programs, include user-definable macros or other programming capabilities. The DDE design anticipates the combination of these applications based entirely on user programming. If the application implements a substantive part of DDE capabilities, the user can supply the topics and item names, constructing data relationships not contemplated by the application developer. Gen-

Figure 6: Foreign Intrigue



eralized applications can thus be combined into tightly coupled customized applications suites.

This process is not as complex as it might sound. In the bar chart example described above, the user can specify, via a dialog box, the particular stocks to be tracked. In a more generalized implementation, the user might also have been allowed to specify the topic, directing the DDE message to a different server. If, hypothetically, another server existed in the PC, it could supply the data to be plotted. Thus one instance of the bar chart program might display stock market data, while another instance displays temperature, pulse, and blood pressure from a server connected to appropriate sensors.

This example may be farfetched in the choice of data to be displayed side by side, but it's reasonable to imagine the development of common analytical software that can be applied to disparate tasks. Microsoft Windows and DDE expand the potential market for analytical software by providing a protocol that is truly application independent.

Outside Influence

So far this discussion has been limited to communications among applications running on the same PC. The principles of DDE are, however, applicable to transfers of data across machines, even computers of different types.

The DDE protocol assumes that (if possible) a request for data exchange will be satisfied by "broadcasting" a request to all other tasks in the PC and that one of those contacted will supply the desired data. If the data desired is not available locally, DDE provides for the use of "agents."

An agent is an application that serves as the importer/exporter of data. When an agent application receives a request for data about a given topic, it determines whether it can satisfy the request by in turn sending the message to its "correspondents" in other machines, which may be able to satisfy the request.

A given agent might use a local area network, micro-mainframe link, or other communications facility to reach its correspondents. The corres-

WHEN AN AGENT APPLICATION RECEIVES A REQUEST FOR DATA ABOUT A GIVEN TOPIC, IT DETERMINES WHETHER IT CAN SATISFY THE REQUEST BY IN TURN SENDING THE MESSAGE TO ITS "CORRESPONDENTS" IN OTHER MACHINES, WHICH MAY BE ABLE TO SATISFY THE REQUEST.

Figure 7: Server's Initialization Procedure

```
typedef struct
{
        WORD    wHandshake;
        WORD    wFormat;
        HWND    hServerWnd;
        char    szData[ MAXDATACHARS ];
} DDEUPDATEBLOCK;

typedef struct
{
        WORD    DDEMode;
        WORD    wFormat;
} DDEADVISEBLOCK;

typedef struct
{
        int     cActive;                    /* Number of active clients */
        ATOM    aSymbol;                    /* Atom identifies symbol   */
        char    szSymbol[ MAXSYMCHARS ];    /* Text of the symbol name  */
        HWND    hwndClient[ MAXCLIENTS ]; /* Handles of active clients */
} HOTITEMS;
static  HOTITEMS        DDE[ MAXHOTITEMS ];
static  DDEUPDATEBLOCK  DDEUpdateBlock;
static  ATOM            aStockInfo;
                /* This atom means DDE_INITIATE is for us */
void    dde_init()
{
        /* Atoms for WM_DDE_INITIATE */
        aStockInfo = DDEAddAtom( (LPSTR) "NYSE.COM.STK" );
                                ⋮
{
```

Figure 8: Server's INITIATE Response Procedure

```
void    dde_initiate( hwnd, hwndClient, lData )
    HWND    hwnd;
    WORD    hwndClient; /* Contains client's window handle */
    DWORD   lData;
    {
            ATOM    atom;
             atom = (ATOM) HIWORD( lData );
             if( atom == aStockInfo )  {
                    if( IsWindow( (HWND) hwndClient ) )
                            SendMessage( (HWND) hwndClient,WM_DDE_ACK,
                                         hwnd, MAKELONG( TRUE, atom )  ):
    )       }
```

pondent agent may then issue a DDE INITIATE (if it is located in a PC) to locate the ultimate server. The exchange of data then proceeds without either the client or the server being aware of the agents.

Access Control

While DDE does not explicitly provide for access control or data security, a server application has a number of alternatives at its disposal to validate the authority of a given client before exchanging data with it. The server could require a password or other access parameter to be sent along with the topic, transmit a REQUEST message back to the client for the password, or using the window handle provided, access a lot of information (from Windows) about the client.

The DDE protocol is a framework on which software developers can construct private or public exchange mechanisms. When exchanging data with private (well-known) clients, servers can supplement DDE with message priority, data queueing, and private message protocols. In addition to Microsoft Windows' built-in facilities for inter-application communication, very complete communication facilities designs can be implemented.

The Future

The fact that Microsoft Windows is independent of any physical device permits applications to take full advantage of the capabilities of a wide variety of current and future displays, printers, and pointing devices. Application developers are freed from having to

be concerned in their programs with the peculiarities of particular devices. DDE extends the same concept to data exchange. If an application honors DDE requests, it's then able to communicate with all other applications that do so, without having to consider their peculiarities.

DDE is an important step forward in inter-application communication, since it provides a public protocol that applications can subscribe to rather than requiring each application developer to have to consider all the possible applications that users might want to exchange data with. The user can then combine programs and achieve results that no single application can offer. ■

Figure 9: Server's Response to Advise Request

```
void    dde_advise( hwnd, hwndClient, lData )
  HWND    hwnd;
  WORD    hwndClient; /* Contains client's window handle */
  DWORD   lData;
  {
          ATOM                        atom;
          BOOL                        b;
          GLOBALHANDLE                gh;
          DDEADVISEBLOCK FAR *        lpDDEAdviseBlock;
          int                         i, j, k;
          LPSTR                       lps;
          char *                      p;
          char                        sz[ 20 ];
          b = FALSE;
          atom = (ATOM) HIWORD( lData );
                  /* Lookup Symbol in Table */
          if( SymbolIsAvailable( atom ) )  {
              gh = (HANDLE) LOWORD( lData );
              lpDDEAdviseBlock = (DDEADVISEBLOCK FAR *) GlobalLock( gh );

              if( lpDDEAdviseBlock->wFormat == CF_TEXT ) {
                    for( i = 0; i < MAXHOTITEMS; i++ ) {
                        if( DDE[i].aSymbol == atom ) {
                            for( j = 0; j < MAXCLIENTS; j++ )  {
                               if( DDE[i].hwndClient[j]==(HWND)hwndClient)
                                   b = TRUE; /*Already there */
                                   break;
                            }  }
                        if( !b )   /* if atom is there but client isn't */
                        {   for( j = 0; j < MAXCLIENTS;j++ )
                            {         /* Look for an empty slot */
                               if( DDE[i].hwndClient[j]==(HWND)NULL )
                               { DDE[i].hwndClient[j]= (HWND)hwndClient;
                                    DDE[i].cActive++;
                                    b = TRUE;
                                    break;
                    }   }   }  }
                    if( b ) break;
              }
              if( !b )          /* If atom is not already in the table */
              {  for( i = 0; i < MAXHOTITEMS; i++ )
                 {  if( DDE[i].cActive <= 0 )
                    {   DDE[i].aSymbol = atom;
                        DDEGetAtomName( atom, (LPSTR) DDE[i].szSymbol,
                                        MAXSYMCHARS - 1 );
                        DDE[i].szSymbol[MAXSYMCHARS - 1 ] = '\0';
                        DDE[i].hwndClient[ 0 ] =(HWND) hwndClient;
                        DDE[i].cActive = 1;
                        b = TRUE;
                        break;
              }  }  }
                  GlobalUnlock( gh );
          }
           /* Acknowledge one way or the other */
          if( IsWindow( (HWND) hwndClient ) )
                  SendMessage( (HWND) hwndClient,
                          WM_DDE_ACK,
                          hwnd,
                          MAKELONG( b, atom )
                          );
          if( b )
          {
                  /* Transmit latest data we received */
                  if(  ( gh = GlobalAlloc( GMEM_MOVEABLE,
                                           (DWORD) sizeof(DDEUPDATEBLOCK)))
                    && ( lps = (LPSTR) GlobalLock( gh ) ) )
                  {   for( p = (char *) &DDEUpdateBlock, k = 0;
                           k < sizeof( DDEUPDATEBLOCK );
                           *lps++ = *p++, k++ );
                      GlobalUnlock( gh );
                       PostMessage( (HWND) hwndClient, WM_DDE_DATA, hwnd,
                                    MAKELONG( gh, atom ) );
          }       }
          return;
  }
```

GE's GENIE On-line Network Offers Expert Support for Microsoft Windows Developers

General Electric Information Services' worldwide teleprocessing network is offering 24-hour on-line support for Windows software applications developers through **GENIE** (General Electric Network for Information Exchange). Access to GENIE and its variety of GE computer support services is through PCs and terminals via a local phone line available in 450 U.S. cities.

The keystone of GENIE is the Microsoft Windows Developers RoundTable, an electronic interchange for Windows developers. The RoundTable is sponsored and maintained by Microsoft and allows applications developers to scan posted messages on the most recent Windows applications developments, access and download technical updates on Windows, and electronically discuss with experts and other developers ideas on multitasking, visual interface, and hardware independence capabilities.

GENIE's Developers RoundTable also includes such topics as Business-Band Real Time Conferencing, electronic bulletin boards, and a downloadable library of software programs.

Additionally, GENIE supplies a full-service electronic mail system, GE Mail and current news about the personal computing industry through the ISIS High Tech News Wire, the Networker's Journal, and Computing Today! through GENIE's News and Commentary.

Requirements: IBM PC or compatible computer, asynchonous modem (either 300 or 1200 baud), communication software package, telephone.

Current subscription rates for GENIE are $18.00 one-time enrollment charge, $35.00 per connect hour during prime time (8 a.m. to 6 p.m. daily, including weekends and holidays), $5.00 per connect hour during nonprime time (6 p.m. to 8 a.m. daily, including weekends and holidays).
For further information about GENIE, call (800) 638-9636.

New Intel Graphics Coprocessor Makes Windows Sparkle

Joe Desposito

A new graphics coprocessor from Intel, the 82786, is expected to substantially enhance the performance of Microsoft Windows and other graphics applications on the IBM PC, XT, AT, and compatibles. The chip performs near-instantaneous screen updates, draws graphic primitives such as lines and circles at more than 2 million pixels per second, and creates text at a rate of 25,000 characters per second. It also supports multitasking and ultra-high-resolution displays.

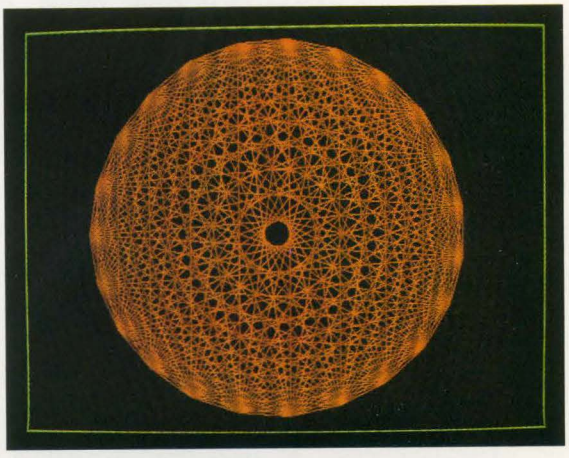
The 82786 has four major components: a graphics processor, display processor, bus interface unit, and dynamic RAM controller. These parts of the chip communicate over a built-in asynchronous bus.

The graphics processor accepts commands from applications programs to draw bit maps, which create objects in memory. The processor contains a set of on-chip graphics commands that draw points, lines, rectangles, circles, polygons, and other primitives, to relieve the host CPU of the task of creating these figures from scratch. To display any of these figures on the screen, the CPU sends a single 16-bit request to the 82786 graphics coprocessor.

To handle specialized drawing functions, the 82786 includes commands such as BitBLT (bit block transfer), ChaBLT (character-block transfer), Incremental Point, and Fill. BitBLTs are performed upon request from the CPU at a rate of 24 megabits per second. ChaBLTs occur at the rate of 25,000 characters per second. The Incremental Point command is used to draw more complex figures. An incremental point list describes the figure's shape, and, once the CPU sends the list to the 82786, the graphics processor draws the shape at a rate of 2 million pixels per second. The Fill command instructs the graphics processor to fill in any shape with horizontal lines.

The coprocessor also has non-drawing instructions. For example, the Link command executes an unconditional jump to another command, and Enter Macro and Exit Macro are used to call and return, respectively, from a subroutine.

The 82786 can receive a



THE CHIP DRAWS GRAPHIC PRIMITIVES SUCH AS LINES AND CIRCLES AT MORE THAN 2 MILLION PIXELS PER SECOND.

Features

- High-performance graphics
- Fast polygon and line drawing
- High-speed character drawing
- Advance DRAM controller for graphics memory up to 4 megabytes
- Supports up to 200-MHz CRTs —up to 640 by 480 by 8 bits (1K•1K•8) or 1400 by 1400 by 1 bit (2K•2K•2)
- Up to 1,024 simultaneous colors
- Integral DRAM controller, shift registers, and DMA Channel
- IBM PC Computer Graphics Adaptor compatbility
- Hardware windows
- Fast bit-block copies between system and bit-map memories
- Integral video DRAM support —up to 1900 by 1900 by 8
- Multitasking support
- International character support
- Provides support for rapid filling with patterns
- 88- pin grid array leadless chip carrier
- Advanced CHMOS technology

block of commands at one time from the CPU, and do it while executing other commands.

Many of the graphics processor's functions are implemented in hardware rather than software. These include clipping, mouse support, character-set support, logical operations, and bit-plane masking.

The amount of display memory is the only limitation on the number of character sets that the 82786 can support. Any one can be activated by a single command. All character sets can have proportional spacing or kerning.

The next major functional block of the 82786 is the display processor. This part of the 82786 is independent of the graphics processor. Microsoft Windows would use the display processor to control display contents.

The display processor transfers multiple bit maps from memory to window areas on the screen. It also provides a hardware cursor and colored borders around the windows. An almost unlimited number of windows can be displayed on the screen.

To create windows, the display processor divides the screen into strips and tiles. A strip is one or more horizontal scan lines, and a tile is a section of a strip. A collection of tiles forms a window.

Unlike the graphics and display processors, the bus-interface unit and dynamic RAM controller are invisible to the CPU software. The bus interface unit maintains the integrity of a display by arbitrating requests for graphics memory. The dynamic RAM controller handles up to 4 megabytes of graphics memory and screen rereshing.

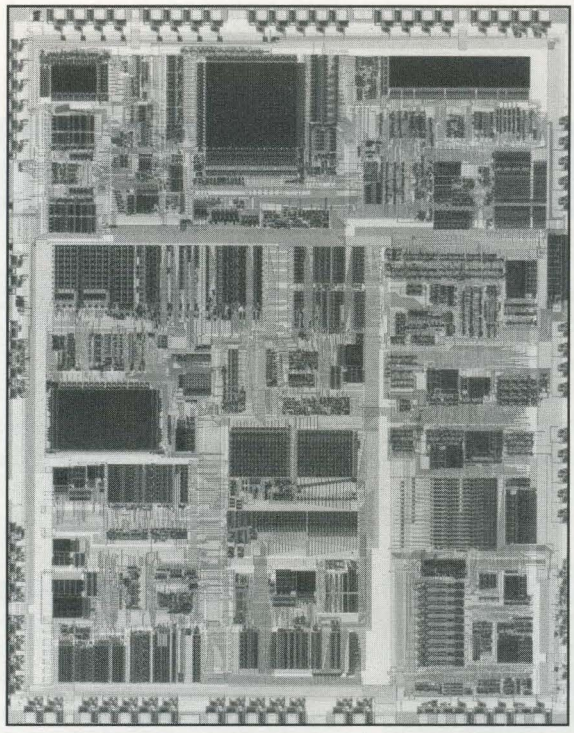
82786 and Windows

What will the 82786 do for a product like Microsoft Windows? According to Richard Bader of Intel, "The 82786 will make Windows so responsive that a computer like the IBM PC will no longer stand in the way of what the user wants to do. Operations that took seconds to execute previously will now happen instantaneously. The 82786 enhances the intimacy between user and machine, and thus enhances productivity."

Will Windows developers need to understand the intricacies of the 82876 in order to achieve this high performance? Says Bader, "When an 82786 board-level product comes to market, it will have a Windows driver, which would then make the 82786 virtually transparent to the applications programmer—as transparent as the Microsoft Graphics Device Interface allows. When an applications programmer uses any graphics board, he needs to interrogate the GDI interface to find out specific characteristics of the board such as the resolution it supports and whether or not it supports color. The application is then adapted to those physical characteristics. But for the most part there is a great deal of transparency provided by GDI, and this will be applicable to new 82786 graphics boards."

An Ideal Application

One application that stands to gain significantly from the appearance of 82786 boards is desktop publishing. Intel's Bader



points out, "As Windows evolves, we expect that it will be ideally suited for desktop publishing. And since the 82786 is so well suited as a GDI device, we believe that the 82786 is also applicable to that market." Many of the desktop publishing packages currently on the market take quite a long time to reformat information on the screen. And displaying a variety of fonts on the screen is time consuming and awkward. Continues Bader, "The speed of the 82786 will change all that. Microsoft Windows in conjunction with the 82786 will give a good performance."

The 82786 supports a wide range of monitors, which is also a plus for desktop publishing. It easily supports systems that are 640 by 480 with 64 colors. Additionally, Bader points out, "We can also support large-format screens, full-page displays, with a resolution in the order of 1200 by 800, 1400 by 1400, or 1000 by 800 with gray scales. What we've found in a number of instances is that people are looking for that 17-inch high-resolution monochrome monior. They're looking for extremely crisp, stable, high-performance text displays, and that's exactly what the 82786 is capable of providing."

Intel and Microsoft have entered a cooperative program to provide Microsoft Windows software driver technology for systems designs using the Intel 82786. ■

Graphics Instruction

64	Bit-Block Transfer Within Bit-Map
AE	Bit-lock Transfer Across Bit-Maps
06	Define Texture: Opaque
07	Define Texture: Transparent
0A	Define Character Set: Word
0B	Define Character Set: Byte
1A	Define Bit-Map
3D	Define Colors
41	Define Logical Combination
46	Define Clip Rectangle
4D	Define Inter character Space
4E	Define Character Orientation
68	Draw Arc: Exclusion
69	Draw Arc: Inclusion
53	Draw Point
B4	Draw Incremental Points
BA	Draw Series of Horizontal Lines
54	Draw Line
58	Draw Rectangle
73	Draw Polygon
74	Draw Polyline
8E	Draw Circle
A6	Draw Character String: Opaque
A7	Draw Character String: Transparent
44	Enter Pick Mode
45	Exit Pick Mode
4F	Move to Absolute Position
52	Move to Relative Position

WINDOWS TRAINING SEMINARS

Corporate software developers can receive introductory and advanced hands-on instruction on Windows applications by participating in Windows development classes offered through Boeing Computer Services.

Microsoft Windows--Overview (Course No. 005)

This two-day course is designed for project managers, application development managers, application designers, and programmers. Attendees should have experience in application development software. Through lecture, instructor-led demonstrations, and student hands-on exercises, the course is a comprehensive introduction to the Microsoft Windows environment. Subjects covered include the use of Windows, the impact of a visual interface, resource management, use of the mouse, and basic application development techniques.

Microsoft Windows--Application Programming Workshop (Course No. 006)

A five-day, hands-on workshop, this course is intended for software designers and programmers who are interested in developing or converting applications to be used in the Microsoft Windows environment. To take advantage of the many festures of Microsoft Windows, applications must be designed and written using the Windows Development Tooolkit. This course covers the toolkit, structure of Windows' application, and steps involved in builing the application. Team exercises are used to lead attendees through the capabilities and complexities of Windows programming. The C programming language is used for all exercises. Prerequisites for the course are successful completion of UNIX—C Programming Language (Course No. 397), IBM PC/XT Hardware and Advanced DOS Concepts (Course No. 166), and Microsoft Windows—Overview (Course No. 005), Applicable work experience may be substituted for the C Programming.

Boeing Computer Services also offers its Windows training programs to developers on an on-site basis.

For additional information and scheduling, contact:

Boeing Computer Services
Education and Training Division
National Training Center
375 Corporate Drive South
Tukwila, Washington 98188.

Call, toll free, (800) 342-7700, or in Washington state, call collect, (206) 575-7700.

Programmable Graphics Processor from TI Perks up PC Pixels

Joe Desposito

Texas Instruments® has developed the first graphics microprocessor, the TMS34010. This is a significant development in the world of graphics chips, which to date has been populated with less powerful graphics coprocessors and controllers. The TMS34010 is a 32-bit processor that executes 6-million instructions per seconds (MIPS), addresses up to 4-gigabytes of memory, and includes special pixel-processing instructions in addition to a general-purpose instruction set.

The chip, which will enhance all graphics applications, is of particular interest to developers using Microsoft Windows on the IBM PC and compatibles, where up until now performance has been less than spectacular. The TMS34010 will be contained in an add-on board that will handle all the graphics processing, replacing the monohrome color graphics or enhanced graphics display adaptor.

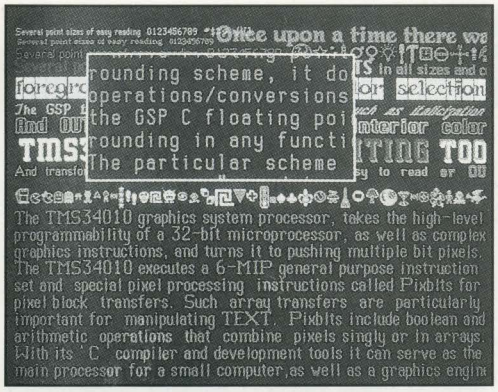
Inside the TMS34010

Like most microprocessors, the TMS34010 has an ALU, general registers, stack pointer, program counter, instruction decoder, and control unit. However, it also contains components that optimize the chip for graphics processing.

It has a far greater number of general registers contained in the microprocessor than that found in general-purpose processors. Thirty 32-bit registers are divided into an A file and a B file. This large number prevents the shifting of bits between registers that often occurs with graphics processing. It uses a barrel shifter for high-speed bit rotations and can rotate a field of up to 32-bits in a single instruction cycle.

The microprocessor uses a four-segment instruction cache for loading instructions from memory before they are executed. Each segment has eight subsegments, each holding eight bytes of code. This 256-byte instruction cache speeds up the execution of graphics algorithms by capturing loops of instructions. Once a loop is captured, "fetch" memory cycles in the loop halt. Loop execution is fast because the chip can access instruction

THE TMS34010 IS A 32-BIT MICROPROCESSOR THAT EXECUTES 6 MILLION INSTRUCTIONS PER SECOND, ADDRESSES UP TO 4 GIGABYTES OF MEMORY, AND INCLUDES SPECIAL PIXEL-PROCESSING INSTRUCTIONS IN ADDITION TO A GENERAL-PURPOSE INSTRUCTION SET.


Several point sizes of easy reading 0123456789
Once upon a time there wa
rounding scheme, it do
operations/conversions
the GSP C floating poi
rounding in any functi
The particular scheme
The TMS34010 graphics system processor, takes the high-level programmability of a 32-bit microprocessor, as well as complex graphics instructions, and turns it to pushing multiple bit pixels. The TMS34010 executes a 6-MIP general purpose instruction set and special pixel processing instructions called PixBlts for pixel block transfers. Such array transfers are particularly important for manipulating TEXT. PixBlts include boolean and arithmetic operations that combine pixels singly or in arrays. With its 'C' compiler and development tools it can serve as the main processor for a small computer,as well as a graphics engine

THE TMS34010 USES HIGH-SPEED INSTRUCTIONS CALLED PIXBLTs (PIXEL BLOCK TRANSFERS) TO HANDLE CHARACTER DISPLAY AND BIT-MAPPED GRAPHICS. PIXBLT INSTRUCTIONS INCLUDE BOOLEAN AND ARITHMETIC OPERATIONS, EXPANSION OF A SINGLE BIT TO ONE OF TWO COLORS, TRANSPARENCY DETECTION, AND PLANE-MASKING.

cache, internal register files, and external memory simultaneously instead of sequentially.

The chip's input/output registers include interrupt registers, host interface registers (for communicating with a host processor), video timing registers (to drive a CRT), and local memory control registers.

To further increase speed, the TMS34010 employs some of the principles of reduced-instruction-set computer (RISC) chips; for example, it executes most of its instructions in a single 160-ns cycle.

Pixel Processing

In addition to approximately 100 general-purpose instructions, the TMS34010 uses special high-speed instructions called PixBLTs (pixel block transfers) to handle character display and bit-mapped graphics. These instructions are an extension of BitBLTs (bit-block transfers) found in some graphics controllers. Transfers are made between source and destination arrays using linear addressing or X-Y coordinates.

PixBLT instructions include Boolean and arithmetic operations, expansion of a single bit to one of two colors, transparency detection, and plane-masking. These types of instructions can also be combined to perform a much larger set of operations.

The TMS34010 operates on single-bit pixels, which are usually associated with monochrome graphics, by using its Boolean operations on source and destination arrays. The common combining operations are Replace, OR, NOT-AND, AND, and XOR.

For multiple-bit pixels, which are usually associated with color and gray-scale graphics, the processor uses arithmetic operations such as Maximum, Minimum, Add, Add with Saturate, Subtract, and Subtract with Saturate. The Maximum and Minimum functions are used to create a sense of depth and to smooth jagged lines, a technique called anti-aliasing. Add with Saturate and Subtract with Saturate are useful for dedicating multiple color planes to each of the three primary colors.

A PixBLT instruction can expand a bit to one of two colors by taking one bit per pixel data from memory, expanding it to multiple-bit color data, and placing it anywhere in the frame buffer. PixBLT instructions also handle windowing, allowing the placement of windows of variable size and content anywhere on the display, and clipping, which prevents drawing outside the boundaries of a window.

The way the TMS34010 handles text generation is quite different from other processors. First, a display list, which defines what is to be displayed, appears. For normal block text, the list is simple; say, a list of ASCII character codes. As text becomes more complex, requiring, for example, proportional spacing and kerning, the display list becomes more complicated. The TMS34010 general-purpose instructions typically break down the display list, and then a PixBLT is used to move this image from off-screen memory onto the screen.

Texas Instruments is supporting the TMS34010 with a line of software development and

support products, including low-level assembly language tools, a C compiler with a run-time support package, a function library, and support of the CGI (computer graphics interface) standard. Hardware and software development systems are also available.

Building Applications

Since the TMS34010 is a microprocessor, a software developer has a great deal of flexibility in determining what runs on the graphics processor and what runs on the host processor. Since the chip will also support a number of graphics standards such as CGI and Microsoft's GDI (graphics device interface), many applications can be ported by using these standards.

One way that developers can tap the potential of the chip is to port a graphics environment like Microsoft Windows to it at GDI levels. Thus any software application running through Windows could be accelerated by a 32-bit graphics processor.

However, many of the graphics cards being developed are relatively "soft" cards, in that they are RAM based. In effect, the command set for the board will be downloaded to it. As Kevin McDonough of Texas In-

THE DISTINGUISHING FEATURE OF THE TMS34010 IS THAT THE PROCESSOR IS NOT LIMITED TO A FEW ALGORITHMS.

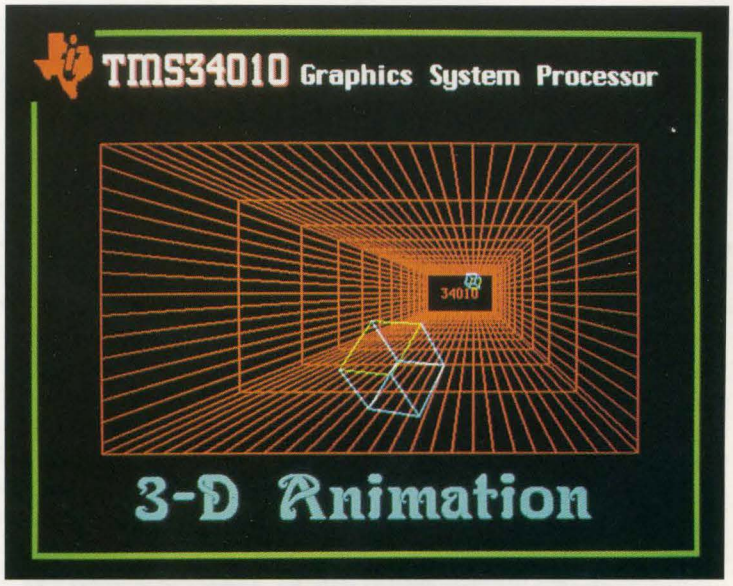
Graphics Instruction

ADDXY	RS,RD	Add Registers in XY Mode
CMPXY	RS,RD	Compare X and Y Halves of Registers
CPW	RS,RD	Compare Point to Window
CVXYL	RS,RD	Convert XY Address to Linear Address
DRAV	RS,RD	Draw and Advance
FILL	L	Fill Array with Processed Pixels: Linear
FILL	XY	Fill Array with Processed Pixels: XY
MOVX	RS,RD	Move X Half of Register
MOVY	RS,RD	Move Y Half of Register
PIXBLT	B,L	Pixel Block Transfer: Binary to Linear
PIXBLT	B,XY	Pixel Block Transfer: Binary to XY
PIXBLT	L,L	Pixel Block Transfer: Linear to Linear
PIXBLT	L,XY	Pixel Block Transfer: Linear to XY
PIXBLT	XY,L	Pixel Block Transfer: XY to Linear
PIXBLT	XY,XY	Pixel Block Transfer: XY to XY
PIXT	RS,*RD	Pixel Transfer: Register to Indirect
PIXT	RS,*RD.XY	Pixel Transfer: Register to Indirect XY
PIXT	*RS,RD	Pixel Transfer: Indirect to Register
PIXT	*RS,*RD	Pixel Transfer: Indirect to Register
PIXT	*RS.XY,RD	Pixel Transfer: Indirect XY to Register
PIXT	*RS.XY,*RD.XT	Pixel Transfer: Indirect XY to Indirect XY
SUBXY	RS,RD	Subtract Registers in XY Mode
LINE	Z	Line Draw

Key to abbreviations: S (source register), D (destination register), R (register file select), K (constant), Z (draw option)

Features

- 160-ns instruction cycle time
- Fully programmable 32-bit general-purpose processor with 128-megabyte address range
- Pixel processing, X-Y addressing, and window clip/pick built into the instruction set
- Programmable 1-,2-,4-,8-, or 16-bit pixel size with 16 Boolean and 6 arithmetic pixel processing options (Raster-Ops)
- 31 general-purpose 32-bit registers
- 256-byte LRU on-chip instruction cache
- Direct interfacing to both conventional DRAM and multiport video RAM
- Dedicated 8/16-bit host processor interface and HOLD/HLDA interface
- Programmable CRT control (HSYNC,VSYNC,BLANK)
- High-level language support
- Full line of hardware and software development tools including a C compiler
- 68-leaded packaging (PLCC)
- 5-volt CMOS technology



struments points out, "This will give the cards tremendous flexibility and tremendous performance advantages. For example, a program such as AutoCAD running under Microsoft Windows will run much faster in the Windows environment. You can probably gain another performance factor of 10 by doing specific algorithms and downloading them to the graphics processor for specific applications."

According to McDonough, graphics boards incorporating the TM34010 processor are currently under development at Video 7, Number 9, and other vendors.

The distinguishing feature of the TMS34010 is that the processor is not limited to a few algorithms. Rather, any algorithm, be it line, ellipse, circle, or whatever, can be programmed. And similarly, for text, the chip can be programmed for any format.

Karl Guttag, also of Texas Instruments, points out that the chip is also excellent at performing data compression algorithms. "You know that when you draw images, they have to be stored on disk. And since the images take up a lot of disk space, it makes sense to compress them. Data compression mostly involves bit manipulation, which the TMS-34010 excels at, so it turns out that this chip is better than any general-purpose processor for doing image compression."

Texas Instruments and Microsoft have entered a cooperative program to provide Microsoft Windowssoftwaredrivertechnology for systems designs using the TMS34010.

With graphics boards that incorporate the TMS34010 processor now under development, a new world of graphics will soon open up to personal computer users. ■

Developers' Aid: Dialog Editor

Charles Petzold

The Microsoft Windows dialog box is both one of the most important elements of a Windows application and the most difficult element to design. Since dialog boxes allow the user to select or enter information too complex for an application's menu, they must be clearly and logically organized. Yet the graphical image of the dialog box is normally derived from a template in a Resource Script (.RC) file, where all the coordinates and sizes are specified as numbers. For the programmer, editing the .RC file to get these coordinates and sizes right (both in terms of clarity and aesthetics) is mostly a matter of trial and error.

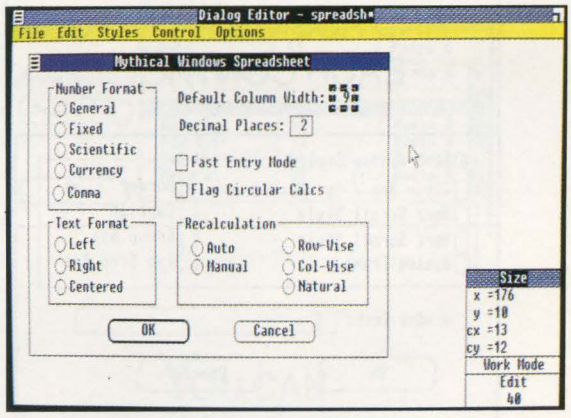
New Dialog Editor

The only help the original Microsoft Windows Software Development Kit offered in designing dialog boxes was the barely adequate DLGEDIT.EXE. If you've tried DLGEDIT and rejected it, don't overlook version 2 of the dialog editor, now called DIALOG.EXE. It's a complete rewrite and a major improvement. A beta test version of the new dialog editor is distributed at Windows Development Seminars and can also be downloaded from the Microsoft SIG on CompuServe.

Aside from it's practical value in helping you create attractive dialog boxes for Windows applications, DIALOG is also an excellent tool to learn about the various styles, controls, and options of dialog boxes. The only problem is that it is distributed without documentation. But like most Windows applications, some creative playtime unlocks most of it's treasures.

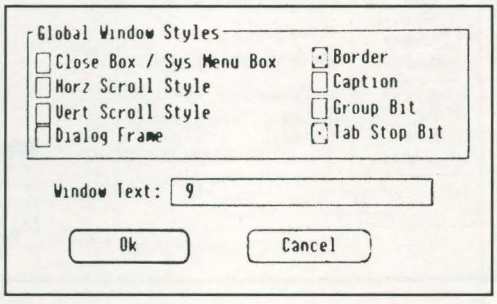
DIALOG saves files in two different formats: a .RES format (the compiled format that it uses for later editing) and a human-readable ASCII format with the extension DLG. The .DLG file contains the complete dialog template as it would normally appear in a .RC file, with style parameters expressed in terms of the macro definitions from WINDOWS.H (such as BS_PUSHBUTTON) and IDs identifying each control as specified in a header file for the application.

One .RES file created by DIALOG can contain many different dialog templates for the application. You can begin a new file within DIALOG by



THE NEW DIALOG EDITOR IS DISTRIBUTED AT WINDOWS DEVELOPMENT SEMINARS AND CAN ALSO BE DOWNLOADED FROM THE MICROSOFT SIG ON COMPUSERVE.

THE STYLES MENU ALLOWS YOU TO FURTHER REFINE THE CONTROLS, REPLACE THE WORD TEXT WITH SOMETHING ELSE, AND ASSIGN AN ID TO EACH CONTROL.



selecting New from the File menu. To start a new dialog template, select New Dialog from the Edit menu and switch between dialog templates by using the View Dialog option of the File menu.

To fill up the dialog box with various controls, use the Control menu, which lists 12 basic types of controls (such as Check Box, Radio Button, Push Button). If the control uses text, it will simply show the word TEXT. You can select the control for editing simply by clicking on it. By hooking the sides or corners you can stretch and size it, and by hooking the center you can move it around.

A Matter of Style

The Control menu lets you select from the basic types of controls. The Styles menu allows you to further refine them, replace the word TEXT with something else, and assign an ID to each control. These IDs are usually macro definitions from the application's header (.H) file. Your source program can then reference each control by these names.

Although DIALOG can read a header file and let you use the name (rather than the number), it asks for the header file only when reading an existing RES file. So, before you get too far, you'll want to save a new file using the Save As option from the Files menu, create a header file with define statements for the IDs, read the file back into DIALOG with File Open, and enter the name of the header file when DIALOG asks for it.

From the Styles menu you have a choice of Class Styles and Standard Styles. The options here let you refine the way the control appears on the screen and how it works.

For instance, you might

Figure 1

```
TRYOUT DIALOG 11, 18, 208, 165
 CAPTION "Mythical Windows Spreadsheet"
 STYLE WS_BORDER | WS_CAPTION | WS_DLGFRAME | WS_SYSMENU | WS_POPUP
 BEGIN
     CONTROL "Number Format"  11, "button", BS_GROUPBOX     | WS_CHILD, 12, 4, 64, 76
     CONTROL "General"     ID_GEN, "button", BS_RADIOBUTTON | WS_GROUP | WS_TABSTOP | WS_CHILD, 16, 16, 44, 12
     CONTROL "Fixed"       ID_FIX, "button", BS_RADIOBUTTON | WS_CHILD, 16, 28, 40, 12
     CONTROL "Scientific" ID_SCI, "button", BS_RADIOBUTTON | WS_CHILD, 16, 40, 53, 12
     CONTROL "Currency"    ID_CUR, "button", BS_RADIOBUTTON | WS_CHILD, 16, 52, 45, 12
     CONTROL "Comma"       ID_COM, "button", BS_RADIOBUTTON | WS_CHILD, 15, 65, 46, 12
     CONTROL "Default Column Width:" 8, "static", SS_LEFT | WS_GROUP | WS_CHILD, 88, 12, 92, 10
     CONTROL " 9" ID_DCW, "edit", ES_LEFT | ES_RIGHT | WS_BORDER | WS_TABSTOP | WS_CHILD, 176, 10, 13, 12
     CONTROL "Decimal Places:" 17, "static", SS_LEFT | WS_CHILD, 88, 28, 66, 10
     CONTROL " 2" ID_DPL, "edit", ES_LEFT | ES_RIGHT | WS_BORDER | WS_TABSTOP | WS_CHILD, 152, 26, 13, 12
     CONTROL "Fast Entry Mode" ID_FEM, "button", BS_AUTOCHECKBOX | WS_TABSTOP | WS_CHILD, 87, 47, 86, 12
     CONTROL "Flag Circular Calcs" ID_FCC, "button", BS_CHECKBOX | WS_TABSTOP | WS_CHILD, 87, 63, 105, 12
     CONTROL "Text Format"   19,  "button", BS_GROUPBOX | WS_CHILD, 12, 84, 64, 52
     CONTROL "Left"        ID_LEF, "button", BS_RADIOBUTTON | WS_GROUP | WS_TABSTOP | WS_CHILD, 16, 95, 28, 12
     CONTROL "Right"       ID_RIT, "button", BS_RADIOBUTTON | WS_CHILD, 16, 108, 48, 12
     CONTROL "Centered"    ID_CEN, "button", BS_RADIOBUTTON | WS_CHILD, 16, 120, 52, 12
     CONTROL "Recalculation" 3, "button", BS_GROUPBOX | WS_GROUP | WS_CHILD, 88, 84, 109, 52
     CONTROL "Auto"        ID_AUT, "button", BS_RADIOBUTTON | WS_GROUP | WS_TABSTOP | WS_CHILD, 96, 97, 28, 12
     CONTROL "Manual"      ID_MAN, "button", BS_RADIOBUTTON | WS_CHILD, 96, 108, 39, 12
     CONTROL "Row-Wise"    ID_ROW, "button", BS_RADIOBUTTON | WS_CHILD, 148, 96, 47, 12
     CONTROL "Col-Wise"    ID_COL, "button", BS_RADIOBUTTON | WS_CHILD, 148, 108, 45, 12
     CONTROL "Natural"     ID_NAT, "button", BS_RADIOBUTTON | WS_CHILD, 148, 119, 44, 12
     CONTROL "OK" 0,      "button", BS_DEFPUSHBUTTON | WS_GROUP | WS_TABSTOP | WS_CHILD, 48, 144, 47, 14
     CONTROL "Cancel" 0, "button", BS_PUSHBUTTON | WS_TABSTOP | WS_CHILD, 120, 144, 47, 14
 END
```

select Push Button from the Control menu, which will display text in a rounded rectangle and is often used for things like OK and CANCEL If you want a default setting for the push button (which has a bold outline), you first select the push button control for editing, bring down the Class Styles option of the Styles menu, and select Def Push Button. Another example: The dialog box itself is initially shown with a simple frame. You can add a caption bar and a system menu box by selecting those options from Standard Styles.

Controls in a Windows dialog box are generally organized by "groups" and "tab stops." Controls marked as tab stops can be navigated using the tab key. Within a group, however, you can use the arrow keys. You can mark controls as groups and tab stops from the Standard Styles option of the Styles menu. However, if you've been designing your dialog box in a free-form manner, you'll find that the tab stops will bounce around from item to item in the order that you've created them rather than a visually logical order.

Fortunately, you can both reorder the controls and select groups and tab stops through the Order Groups option of the Options menu. This shows a list of all the controls you've created. Selecting a control with the mouse and placing the horizontal black bar cursor in a different location moves the control. To define a group, you have to mark only the first and last control of the group. You would usually flag the first control of the group as a tab stop also, but you can pick another control as the tab stop if you'd like the cursor to jump to the middle of a group.

Make Sure It Works

Of course, now you'll have to recompile to see if the thing works right. Well, not quite yet. You can now pick Test Mode from the Options menu and test the tab stops and groups from within DIALOG. This test mode is one of the nicest features of the program.

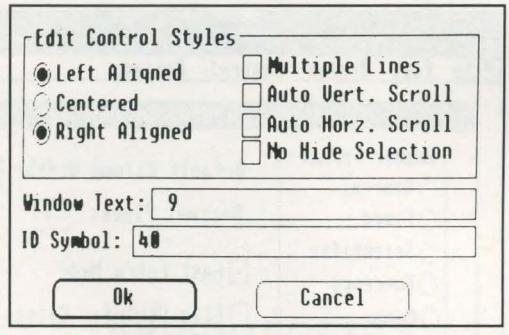
DIALOG is not perfect. You should watch out for a few things: If you want to delete a specific control from the dialog box, first select it and then pick Clear Control from the Edit menu. The Cut Dialog option of the Edit menu deletes the whole dialog box and everything in it. You can get it back by selecting Paste Dialog from the Edit menu, but it's a little disconcerting to see the whole thing disappear.

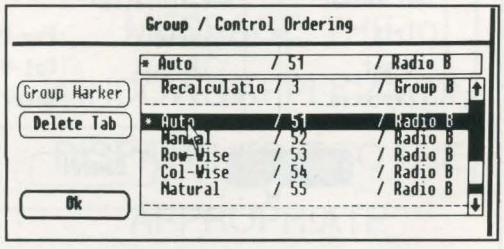
Icons do not work right. You should be able to identify an icon by typing the name in the Text field of the Class Styles box. However, DIALOG will ignore this name. You can either avoid using icons in dialog boxes or enter in the name later using a text editor.

The worst problem does not lie with DIALOG. Preferably, you should be able to include the .DLG file created by DIALOG in the application's Resource Script (.RC) file, as:

```
#include"myprog.dlg"
```

However, the RC.EXE resource compiler version 1.00 will not correctly process an

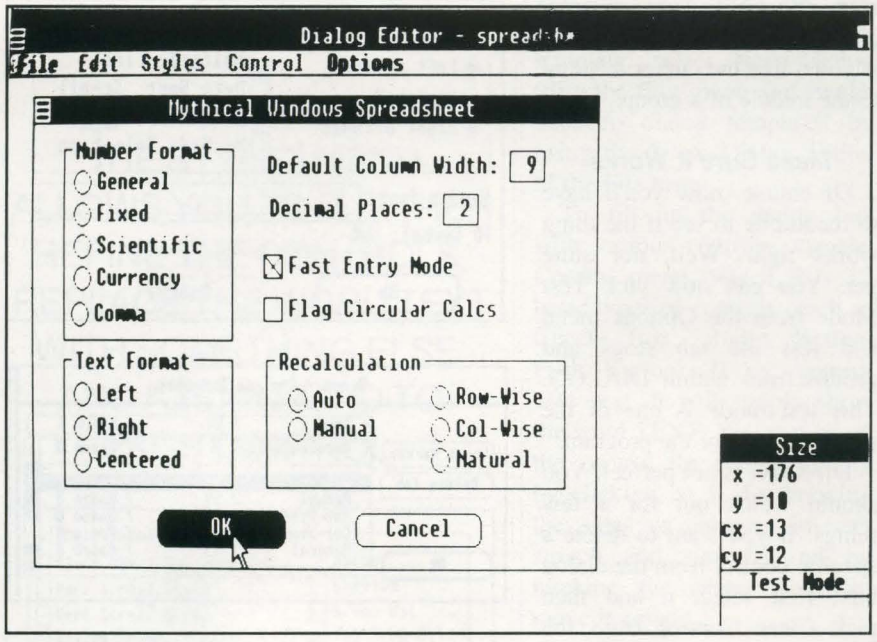



Group / Control Ordering
Group Marker
Delete Tab
Ok
* Auto / 51 / Radio B
Recalculatio / 3 / Group B
* Auto / 51 / Radio B
Manual / 52 / Radio B
Row-Wise / 53 / Radio B
Col-Wise / 54 / Radio B
Natural / 55 / Radio B

YOU CAN REORDER THE CONTROLS AND SELECT GROUPS AND TAB STOPS THROUGH THE ORDER GROUP OPTION OF THE OPTIONS MENU.

Required Reading

Chapter 7 (pages 147 to 174) of the *Microsoft Windows Software Development Kit Programming Guide* (for a general discussion of controls and dialog boxes).
Section 6.2.6 (pages 292 to 307) of the *Programmer's Reference* (for the format of the DIALOG template).



include file in the resource script unless the include file contains only define statements. Until this problem has been fixed, you'll probably have to copy the .DLG file into your .RC file manually. You'll also want to use a different name for the .RES file that is created by DIALOG and used for future editing. Otherwise, DIALOG's .RES file will be overwritten when you run the RC resource compiler. Although DIALOG can read and properly modify a RES file that it did not create and which contains information other than the dialog boxes, it would probably be safer to let DIALOG work with its own .RES file. ■

Microsoft's DIAL Offers Users Electronic Access to Microsoft Customer Support

Microsoft Direct Information Access Line, **DIAL**, is an on-line data service offering Microsoft product users electronic access to Microsoft's Customer Support Engineers. The program provides for the preparation, transmission, and administration of Technical Assistance Requests used to report problems, request information, make suggestions, and receive solutions and answers to inquiries.

In addition, DIAL provides direct acccess to Microsoft electronic mail (email) for communication with account managers as well as access for bulletin boards that have information about known bugs, common questions and answers, application notes by product, as well as other information.

DIAL also provides automatic code updates and the ability to transmit binary and ASCII files along with Technical Assistance Requests.

Requirements: IBM PC or compatible with at least 128K of memory, asynchronous communications adapter (RS232 card) or modem card, 1200- or 300-bps modem connected to a good-quality telephone line.

Additional information about DIAL can be obtained by calling: (206) 882-8080.

Ask Dr. Bob!

Multitasking?

Dear Dr. Bob,
I'm confused: Is Microsoft Windows a multitasking environment or not? Sometimes it seems like it is, and sometimes it seems like it isn't. What's the story, Dr. Bob? —Confused

Dear Confused,
Most people think of multitasking as a process requiring time-slicing. In a traditional multitasking environment, the operating system switches between different tasks by allocating short periods of time to each. Microsoft Windows does not yet time-slice between applications. Windows will probably use time-slicing for multitasking only when MS-DOS also does so.

Instead, Windows has a nonpreemptive multitasking scheduler for running Windows applications (programs written specifically for Microsoft Windows). A Windows application will not be interrupted if it does not want to be interrupted; only when it specifically relinquishes control do other applications get a chance to run.

A Windows application normally sits in a loop waiting to receive messages, usually executing a GetMessage call. The messages it receives cover all the events (keystrokes, mouse movements, window resizing, and so forth) that affect the application. When it gets a message, the Windows application dispatches it to the appropriate procedure in the program. This procedure can either act on the message or ignore it.

Microsoft Windows switches between applications when an application polls for a message by using the GetMessage call or the similar WaitMessage call. Once it has the message, the application can take as long as it wants, without interruption. In the interim other applications (for instance, the CLOCK) will stop working.

Polite Windows applications check frequently for messages while performing lengthy chores. This way they can yield control to other applications and give them an opportunity to run.

However, if a Windows application needs an unbroken chunk of time (say, for high-speed real-time data acquisition), it can take it. The application can interrogate the mouse and the keyboard, write to the screen, and do everything it needs to do—all without interruption. It just has to avoid looking for more messages.

So Microsoft Windows is multitasking, but it's up to the individual Windows applications when to allow multitasking with other applications and when not to.

For "good old applications" (MS-DOS applications that don't write directly to the display and that can run in a

WHEN IT GETS A MESSAGE, THE WINDOWS APPLICATION DISPATCHES IT TO THE APPROPRIATE PROCEDURE IN THE PROGRAM. THIS PROCEDURE CAN EITHER ACT ON THE MESSAGE OR IGNORE IT.

window), Windows yields control to other applications during keyboard input and screen output. But for old applications that do write directly to the display, Microsoft Windows suspends all multitasking but still allows the user (in most cases) to switch among applications.

Compilers

Dear Dr. Bob,
Why do I need to use Microsoft compilers for developing Windows applications?
—Puzzled

Dear Puzzled,
Right now the only compilers that support Microsoft Windows are the Microsoft C and Pascal compilers. You can also program Windows applications in assembler, but the most convenient way to do that would be to use assembler routines called from a C program. Microsoft has made public the relatively minor support for Windows that a C compiler requires, and so other compilers may also support Microsoft Windows.

A C compiler for Windows requires procedures to be "Pascal" (arguments are pushed on the stack from left to right and that the procedure itself adjusts the stack). Normally, C pushes arguments right to left and adusts the stack after returning from the procedure. The C compiler must also allow mixed-model programming with the "near" and "far" key words. Procedures called from Microsoft Windows must have a small prolog and epilog so that Windows can adjust data segment references after moving data.

No Software Interrupt

Dear Dr. Bob,
Is there some kind of software interrupt or other hook I can use to access Microsoft Windows procedures from within a program I write in another language, such as BASIC or Turbo Pascal? Can I get at Windows graphics functions any other way?
— Frustrated

Dear Frustrated,
No. Windows functions are not accessed through a software interrupt. The Windows loader resolves all function references in the new format executable at load time. This means that only programs in the new EXE format can access Microsoft Windows functions.

It might be possible to write a shell program as a Windows application that would provide a path for other programs that are not Windows applications to call Windows procedures. But this task would be very difficult. Moveover, these programs would almost certainly have to be fixed in memory and could not be moved.

The WINOLDAP module allows some old applications to run in a window, by providing one type of shell that intercepts DOS and BIOS calls and translates these calls into Windows functions. This shell is one of Windows' more complex features, since it must deal with many of the nasty things that old applications do. A more extensive shell—even one that included only graphics support—would have to be more complex.

Executive

Dear Dr. Bob:
Can I execute a Windows application from within another Windows application?
—Stumped

Dear Stumped,
Sure. That's what the MS-DOS Executive does. The recommended method for executing a program from a Windows application is with an assembly language subroutine using the EXEC function call 0x4B. Windows intercepts this MS-DOS call and can correctly load the new format executable.

The EXEC function call returns to the calling program right after the child process has been loaded. Under MS-DOS, the calling program doesn't get control back until the child process has terminated.

You can even execute an old application from a Windows application or vice versa. You can get a little sense of what happens in this latter case by running COMMAND.COM from Microsoft Windows. If you lie in the PIF file and say that COMMAND.COMwritesdirectly to the screen, you'll find that executing a Windows application from the MS-DOS command level results in the "This program requires Microsoft Windows" message.

If you run COMMAND-.COM in a window and enter the name of a Windows application, however, it will load correctly and take over the area of the screen occupied by COMMAND.COM. When you

bring up COMMAND.COM again, you'll see the cursor sitting at the MS-DOS prompt. The Windows application has been loaded, and Windows has returned control back to COMMAND.COM.

Expanded RAM

Dear Dr. Bob,
Can Microsoft Windows use an expanded memory board?

—Fuzzy

Dear Fuzzy,

Yes and no. Microsoft Windows can take advantage of a RAM-Disk on an expanded memory board but not of expanded memory itself. However, Windows supports the use of expanded memory by old applications by preserving and restoring the state of the Expanded Memory Manager.

Let's look at these two topics separately.

The presence of any high-speed disk storage will improve Windows' performance. Windows uses the disk for storing temporary files from Windows applications and for swapping old applications in and out of memory. The disk used for these purposes can be any real or virtual disk on your system, including any type of RAM-Disk. A RAM-Disk installed in expanded memory is ideal.

If your RAM-Disk is installed as drive D: (for instance), put the following statement in your AUTOEXEC.BAT:

```
SET TEMP=D:\
```

This statement directs Microsoft Windows applications to store temporary files (for instance, those created by WRITE) on the RAM-Disk. These temporary files begin with a tilde (~) and have the extension TMP.

If you do Windows program development, you should also include the following statement in your AUTOEXEC.BAT:

```
SET TMP=D:\
```

This statement directs the Microsoft C Compiler to use the RAM-Disk for its temporary files.

To use drive D: as a swap-

```
SwapDisk=?
```

with

```
SwapDisk=D
```

ping disk for old applications, replace the line in your WIN.INI file that reads

Beyond using a RAM-Disk in expanded memory, neither Microsoft Windows nor Windows applications use expanded memory directly. (A future version of Windows may have more support for expanded memory.)

However, Microsoft Windows can detect the presence of an expanded memory board. If it finds one, Windows can let old applications use expanded memory in the multitasking environment without memory conflicts. Windows will save the contents of the expanded memory board registers and restore them when switching among applications.

Some early Expanded Memory Managers did not include full support of the "Get/Set Page Map" (function 15) of the Expanded Memory Specification Release 3.20. This call is the one Microsoft Windows uses to save the contents of the expanded memory board registers. For this reason, Release 3.20 versions of the Expanded

WINDOWS CAN TAKE ADVANTAGE OF A RAM-DISK ON AN EXPANDED MEMORY BOARD BUT NOT OF EXPANDED MEMORY ITSELF. HOWEVER, WINDOWS SUPPORTS THE USE OF EXPANDED MEMORY BY OLD APPLICATIONS.

Memory Manager are included with Windows. These drivers are called EMM.PC and EMM.AT; they work only with the Intel Above Boards. You can also use any Expanded Memory Manager, Release 3.20 or later, supplied with an Intel Above Board. Before using any other expanded memory board, check with the manufacturer to determine if the Expanded Memory Manager provides Release 3.20 support of the "Get/Set Page Map" call.

Math Coprocessor

Dear Dr. Bob,
Can Windows applications use an 8087 or 80287 math coprocessor?
—Perplexed

Dear Perplexed,
Yes. Microsoft Windows provides full support for saving the state of the math coprocessor and restoring it when switching among applications. That's the good news. However, Windows applications cannot yet use the 8087 floating point libraries provided with the Microsoft C Compiler (version 3 or 4). This problem lies with the libraries themselves rather than with Windows support of the 8087.

If you want to use floating point in a Windows application written in C, you must currently use the /FPa compiler switch and link with the SLIBFA.LIB, MLIBFA.LIB, or LLIBFA.LIB—depending on your memory model—alternate math library. The alternate math library, which uses a subset of the IEEE standard, is faster and smaller (but less accurate) than the 8087-emulation code of the other floating point math libraries. The alternate math library does not use an 8087 even if one is present.

When the 8087 floating point libraries are eventually modified for Windows, upgrading the program should only require relinking it with LINK4.

In the interim, you could define your own floating point routines that use either the regular C library routines or the assembly language subroutines using the 8087, depending on the presence of the math coprocessor.

Printf

Dear Dr. Bob,
What happens in a Windows application if I try to write to the Display in the conventional way using the C function printf?
—Stymied

Dear Stymied,
Nothing. Microsoft Windows intercepts all DOS and BIOS output headed for the display and dumps it into the bit bucket, where it is never seen again.

If you need to write formatted text output to the screen, you can use sprintf and display the resultant string with Windows TextOut procedure.

Instances

Dear Dr. Bob,
What happens when I run the same program simultaneously in Microsoft Windows? Are the two versions entirely distinct or do they share code and data?
—Unclear

Dear Unclear,
When an application is loaded more than once, the loadings of the application are referred to as different "instances." Generally, multiple instances of the same application share the same code segment but have different data segments. Windows applications also share resources such as menus, icons, cursors, and bitmaps, since such resources are read-only data.

During initialization, an application can determine whether or not it is the first instance of that application. If it is not the first instance, it can retrieve some data that must be shared from the previous instance. ■

The Doctor Is In!

Dr. Bob wants to hear from you. Send him your questions and comments about Windows, DOS, C, Xenix, applications development, romance, life after 40, life after death, love and money, life after marriage, war and peace, life on other planets, the best restaurants, and life after applications development. Address them to Dr. Bob, c/o H. Roark & Associates, 141 E. 44th St., Suite 804, New York, N.Y. 10017.

Microsoft Corporation assumes no liability for any damages resulting from the use of the information contained herein.

Microsoft, the Microsoft logo, MS-DOS, and XENIX are registered trademarks of Microsoft Corporation. Intel is a registered trademark of Intel Corporation. IBM is a registered trademark of International Business Machines Corporation. Lotus is a registered trademark of Lotus Development Corporation. Texas Instruments is a registered trademark of Texas Instruments, Inc.

MICROSOFT®

0000-00-201